16-833: Robot Localization and Mapping, Spring 2024

Homework 2 - SLAM using Extended Kalman Filter (EKF-SLAM)

Jinjiang You (jinjiany)

Due: Friday March 15, 11:59pm, 2024

Your homework should be submitted as a **typeset PDF file** along with a folder including **code only(no data)**. The PDF and code must be submitted on **Gradescope**. In your implementation, please only fill in the functions marked by `TODO`, and play with parameters, and keep other parts unchanged. If you have questions, post them on Piazza or come to office hours. Please do not post solutions or codes on Piazza.

This homework must be done **individually**, and plagiarism will be taken seriously. You are free to discuss and troubleshoot with others, but the code and writeup must be your own. Note that you should list the name and Andrew ID of each student you have discussed with on the first page of your PDF file.

1 Theory (40 points)

1.

$$\begin{aligned}\mathbf{p}_{t+1} &= \begin{bmatrix} x_{t+1} & y_{t+1} & \theta_{t+1} \end{bmatrix}^\top \\ &= \begin{bmatrix} x_t + d_t\cos\theta_t & y_t + d_t\sin\theta_t & \theta_t + \alpha_t \end{bmatrix}^\top \\ &= \mathbf{p}_t + \begin{bmatrix} d_t\cos\theta_t & d_t\sin\theta_t & \alpha_t \end{bmatrix}^\top\end{aligned}$$

2. The uncertainty of the robot's pose at time $t+1$ can also be represented as a 3-dimensional Gaussian distribution $\mathcal{N}(0, \Sigma_{t+1})$, where:

$$\Sigma_{t+1} = G_t\Sigma_t G_t^\top + T_t\texttt{diag}\left(\sigma_x^2, \sigma_y^2, \sigma_\alpha^2\right) T_t^\top$$

$$G_t = \begin{pmatrix} 1 & 0 & -d_t\sin\theta_t \\ 0 & 1 & d_t\cos\theta_t \\ 0 & 0 & 1 \end{pmatrix}$$

$$T_t = \begin{pmatrix} \cos\theta_t & -\sin\theta_t & 0 \\ \sin\theta_t & \cos\theta_t & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

3.

$$\begin{aligned} l_x &= x_t + (r + n_r)\cos(\theta_t + \beta + n_\beta) \\ l_y &= y_t + (r + n_r)\sin(\theta_t + \beta + n_\beta)\end{aligned}$$

4.

$$\beta = warp2pi\left(np.arctan2\left(l_y - y_t, l_x - x_t\right) - \theta_t - n_\beta\right)$$

$$r = \sqrt{\left(l_x - x_t\right)^2 + \left(l_y - y_t\right)^2} - n_r$$

5.

$$H_p = \begin{pmatrix} \frac{\partial \beta}{\partial x_t} & \frac{\partial \beta}{\partial y_t} & \frac{\partial \beta}{\partial \theta_t} \\ \frac{\partial r}{\partial x_t} & \frac{\partial r}{\partial y_t} & \frac{\partial r}{\partial \theta_t} \end{pmatrix}$$

$$= \begin{pmatrix} \frac{l_y - y_t}{(l_x - x_t)^2 + (l_y - y_t)^2} & -\frac{l_x - x_t}{(l_x - x_t)^2 + (l_y - y_t)^2} & -1 \\ -\frac{l_x - x_t}{\sqrt{(l_x - x_t)^2 + (l_y - y_t)^2}} & -\frac{l_y - y_t}{\sqrt{(l_x - x_t)^2 + (l_y - y_t)^2}} & 0 \end{pmatrix}$$

6.

$$H_l = \begin{pmatrix} \frac{\partial \beta}{\partial l_x} & \frac{\partial \beta}{\partial l_y} \\ \frac{\partial r}{\partial l_x} & \frac{\partial r}{\partial l_y} \end{pmatrix}$$

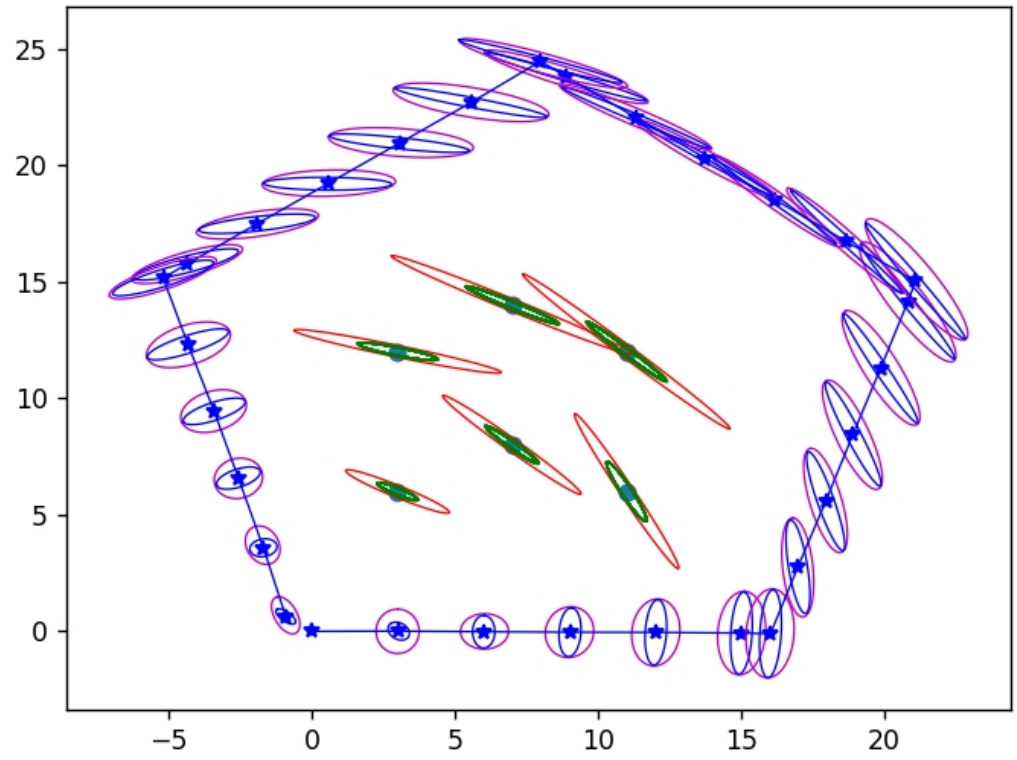
$$= \begin{pmatrix} -\frac{l_y - y_t}{(l_x - x_t)^2 + (l_y - y_t)^2} & \frac{l_x - x_t}{(l_x - x_t)^2 + (l_y - y_t)^2} \\ \frac{l_x - x_t}{\sqrt{(l_x - x_t)^2 + (l_y - y_t)^2}} & \frac{l_y - y_t}{\sqrt{(l_x - x_t)^2 + (l_y - y_t)^2}} \end{pmatrix}$$

We do not need to calculate the measurement Jacobian with respect to other landmarks except for itself because we make the assumption that landmark poses are independent from each other.

2 Implementation and Evaluation (45 points)

In this part you will implement your own 2D EKF-SLAM solver in Python. The robot in problem 1 starts with observing the surrounding environment and measuring some landmarks, then executes a control input to move. The measurement and control steps are repeated several times. For simplicity, we assume the robot observes the same set of landmarks in the same order at each pose. We want to use EKF-SLAM to recover the trajectory of the robot and the positions of the landmarks from the control input and measurements.

1. There are 6 landmarks being observed.

2. 

3. The uncertainties of robot position after motion updates are drawn in magenta, and the those after measurement updates are drawn in blue. We can conclude that motion updates increase the uncertainties of robot position, while measurement updates reduce them.

   The uncertainties of the landmark are drawn in red before SLAM (at initialization stage), and in green during the SLAM process. We can conclude that the EKF-SLAM reduces the uncertainties.

   Overall, the uncertainty ellipses are reduced, meaning that EKF-SLAM is working well to reduce the uncertainty using available information.

4. The figure is already attached above. All of them are inside of the smallest corresponding ellipse. This means that EKF-SLAM predicts the landmark positions well. All the ground truth positions lie within its predicted confidence area.

Landmarks	Euclidean	Mahalanobis
1	0.00271763	0.05242283
2	0.00343081	0.06216337
3	0.00482943	0.03731817
4	0.00538054	0.06432911
5	0.00428718	0.02507875
6	0.00473853	0.09536691

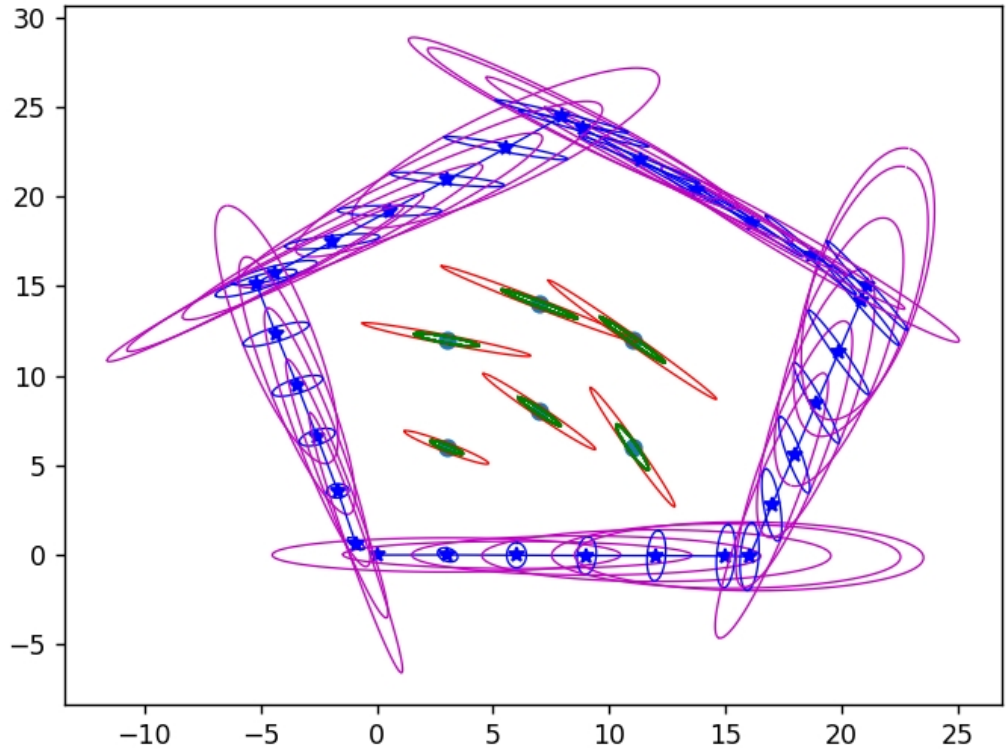
   Both of the distances for each landmark are small, which means the EKF-SLAM predicts landmarks' location well. Meanwhile, The Euclidean distances are smaller than the Mahalanobis distances. In our case, the Mahalanobis distance is a better measurement because it accounts for data distribution and covariance. In our case, we compute the covariance matrix for our predicted state, and the variances along different directions are different. On the other hand, Euclidean distance gives each direction the same weight (isotropic).

3 Discussion (15 points)

1. At the initialization stage, only diagonal blocks in the landmark covariance matrix are non-zero. Other elements are all zero. The zero terms become non-zero in the **update** steps, because the measurements make the landmark positions correlated (The measurements are obtained by the same robot at the same location). Specifically, in the **update** stage we use a for loop to iterate over all landmarks and update the covariance matrix. The matrix $I - K_t H_t$ is not a "block diagonal" matrix. After multiplying this matrix, the covariance matrix is not a "block diagonal" matrix anymore.

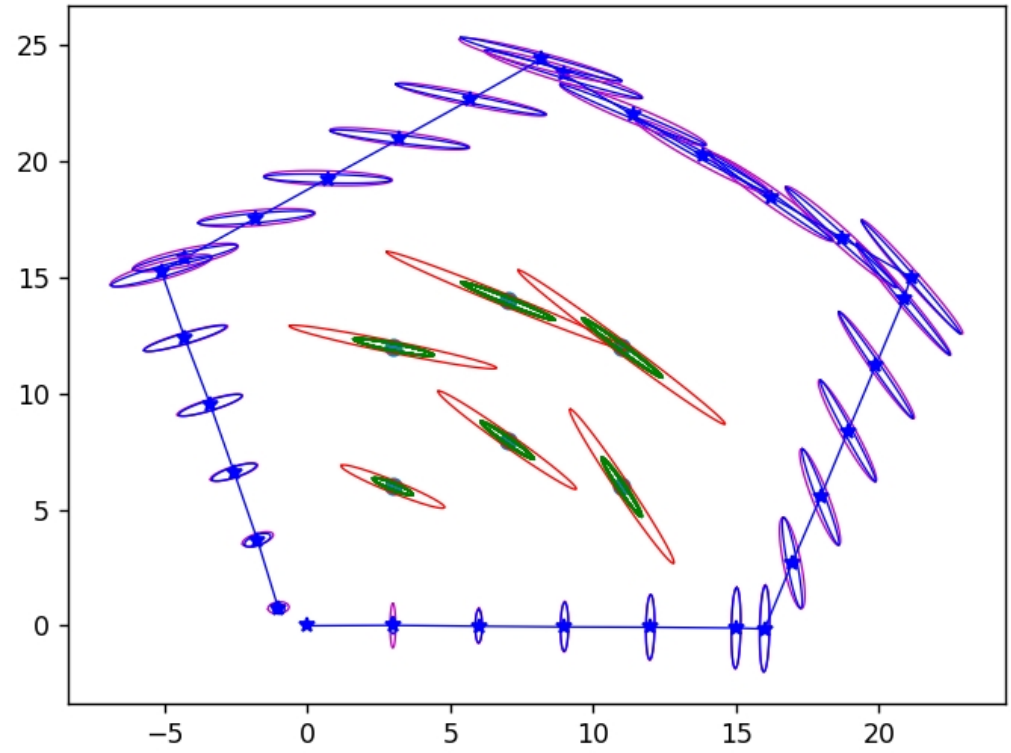
   When setting the initial value for covariance matrix P, we make the assumption that the position of each landmark is independent from each other. However, this is not necessarily true because we are using the measurements of the same frame (i.e. the measurements are obtained by the robot at the same location) to initialize the landmark positions. Therefore, these positions have non-zero covariance.

2. $10 \times \sigma_x$:



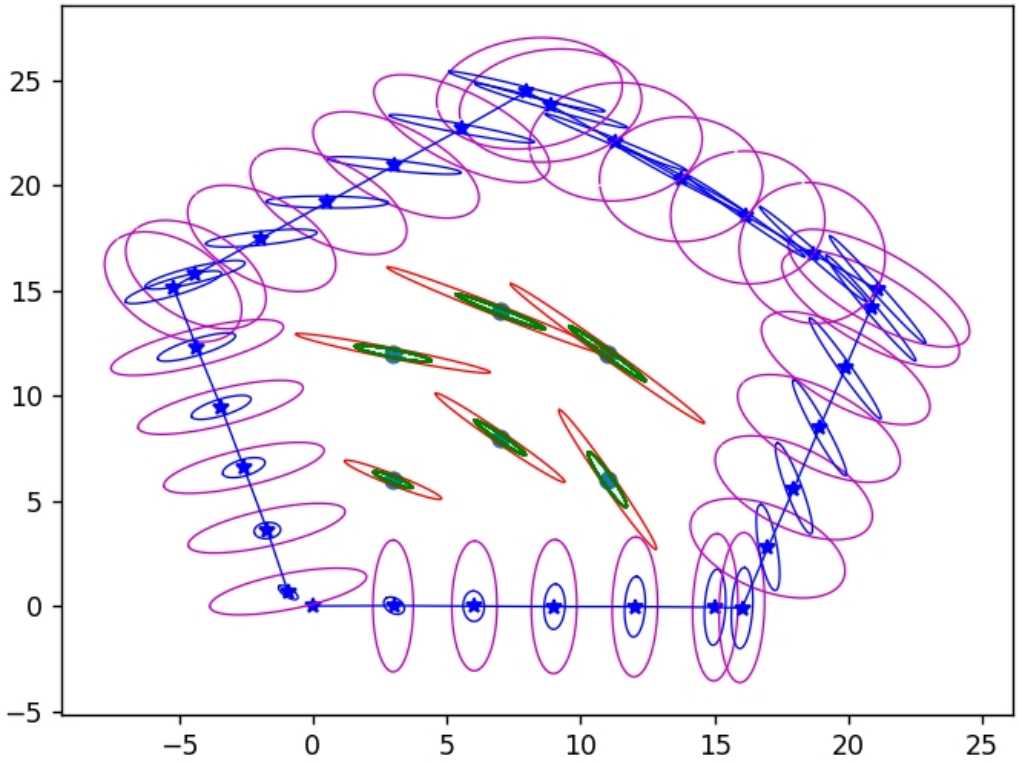
   The covariance of the robot position along the x axis (in robot's local coordinate system) greatly increases. This is because σ_x controls the variance of robot's move information along the x axis (e_x).

   $0.1 \times \sigma_x$:



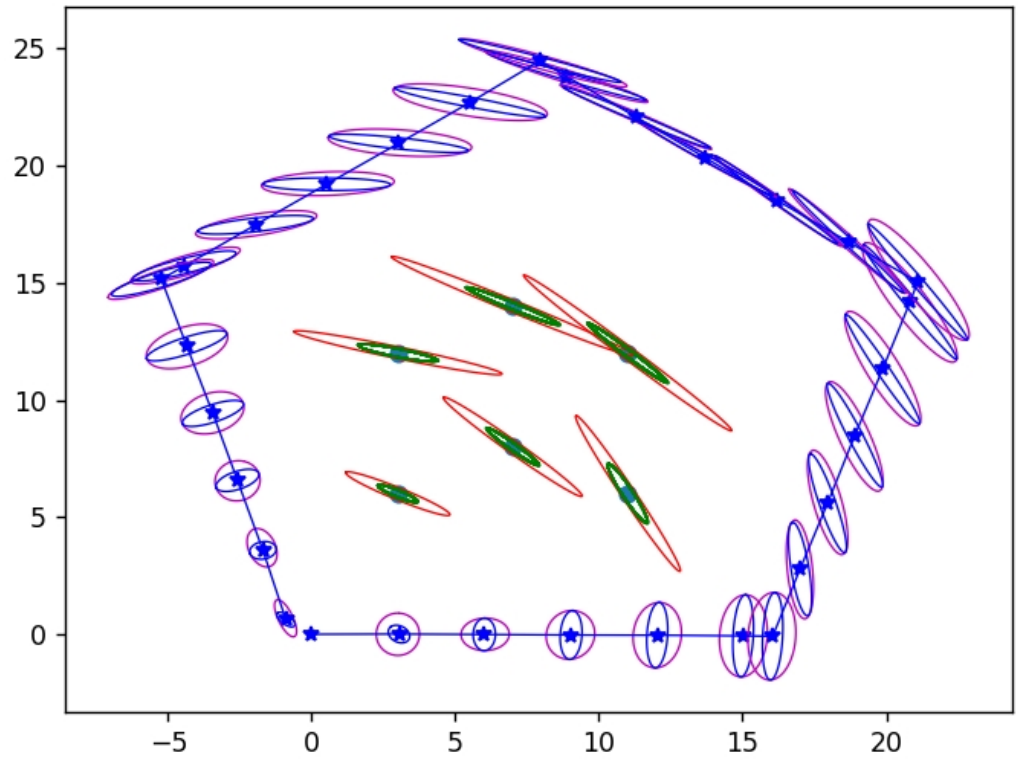
This has the opposite effect.

$10 \times \sigma_y$:



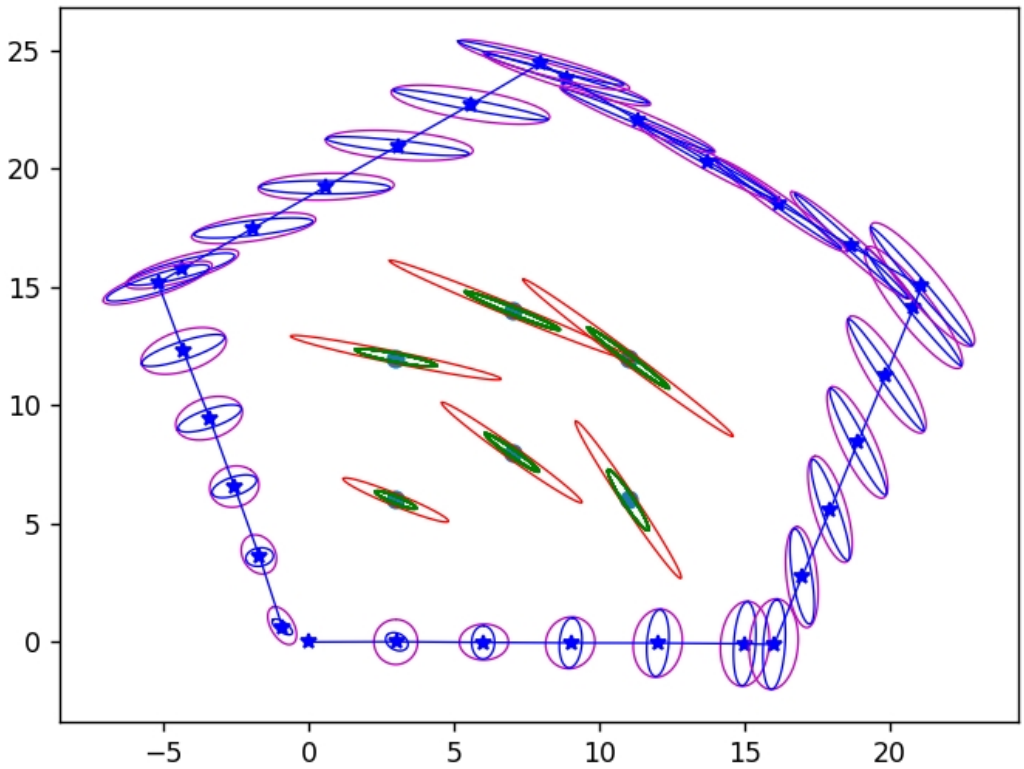
The covariance of the robot position along the y axis (in robot's local coordinate system) greatly increases. This is because σ_y controls the variance of robot's move information along the y axis (e_y).

$0.1 \times \sigma_y$:

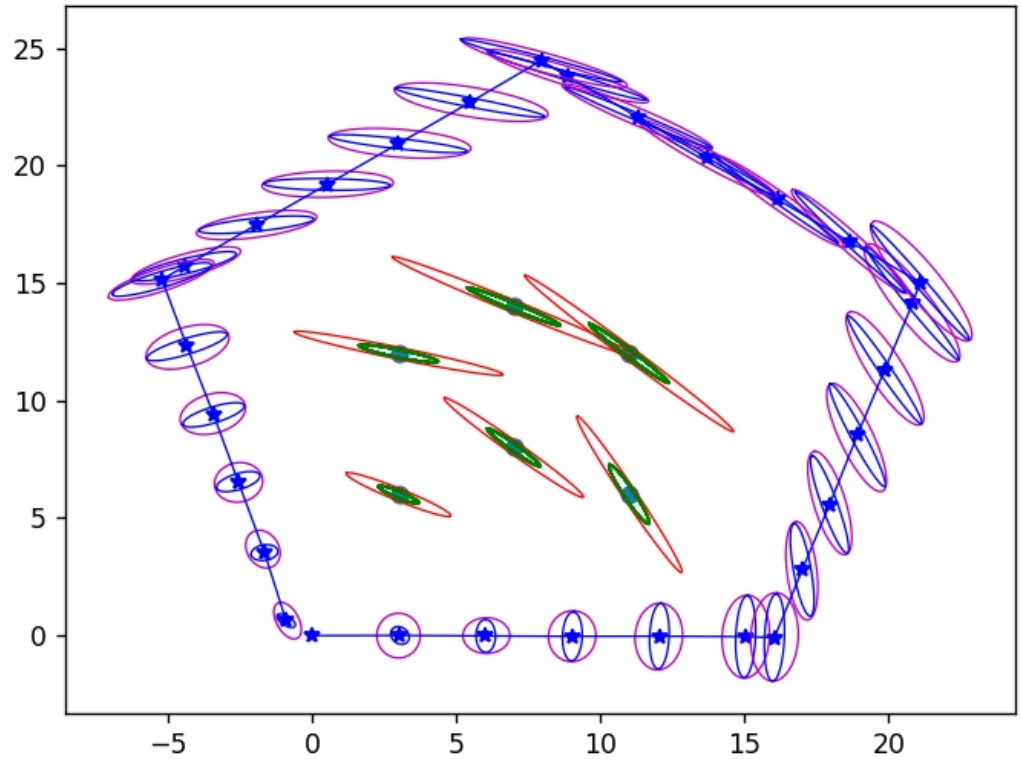


This has the opposite effect.

$10 \times \sigma_\alpha$:

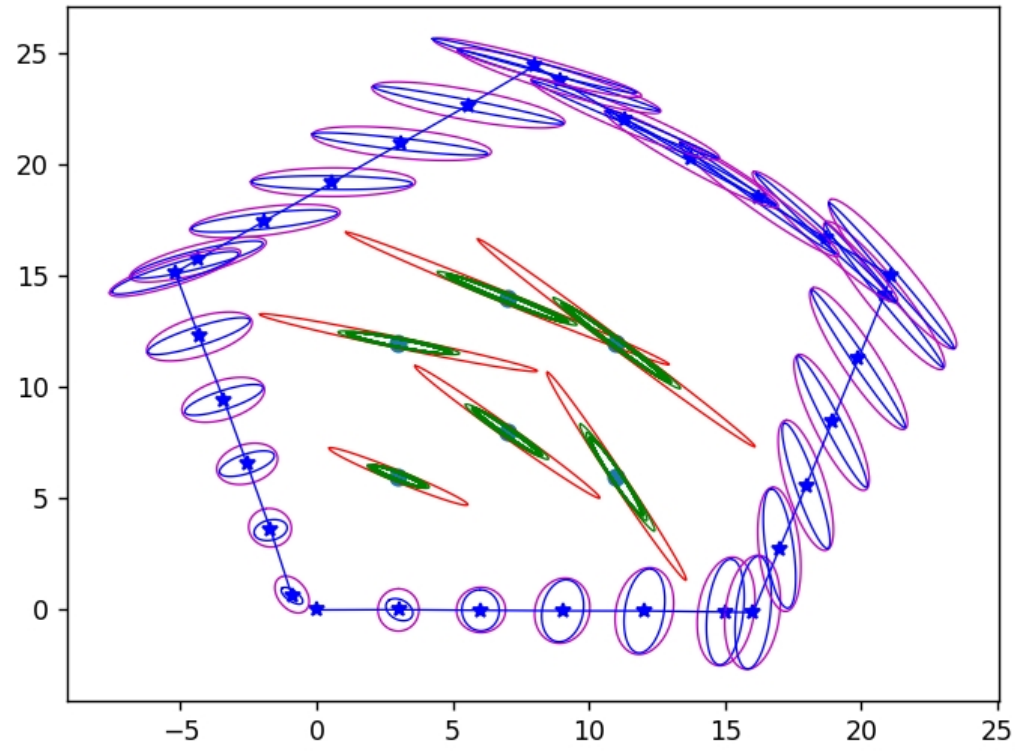


$0.1 \times \sigma_\alpha$:



σ_α controls the variance of robot's rotation information. The difference between different σ_α values is very small. This is because the update step using the measurements of landmarks can greatly reduce the variance of robot orientation (compared with translation).

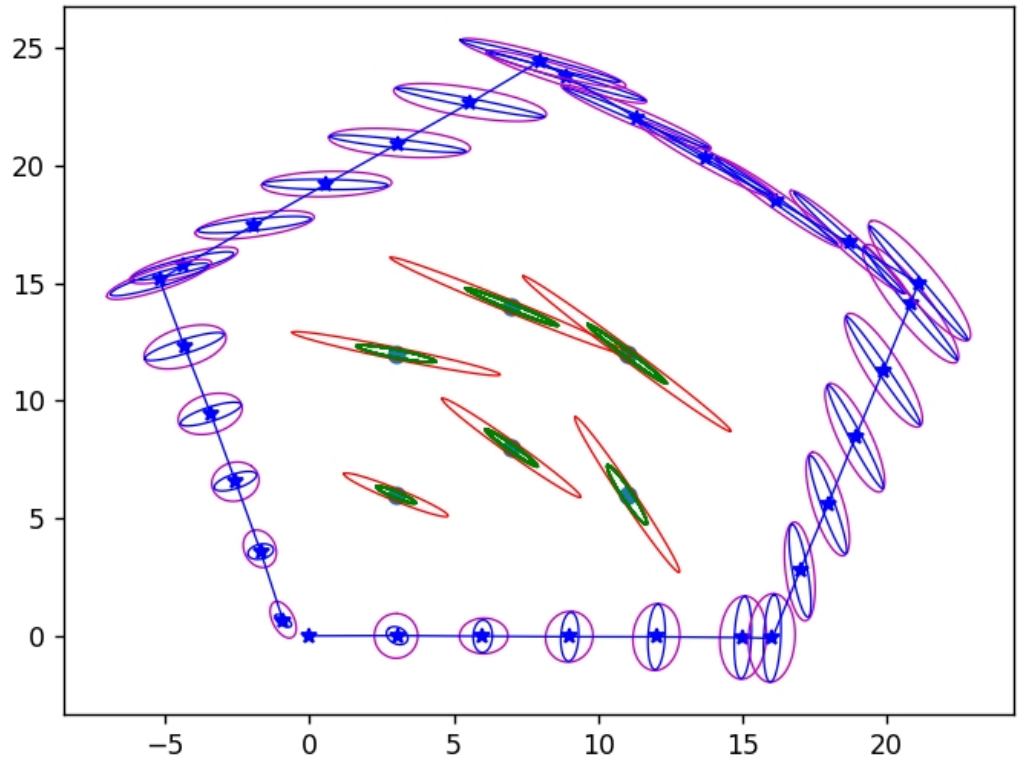
$10 \times \sigma_\beta$:



σ_β controls the variance of the direction from the robot to the landmark. When this value increases, the length of landmarks' ellipses' major axes become longer. In turn, the uncertainty of robot's position also increases because the measurements in update step will influence robot's position.

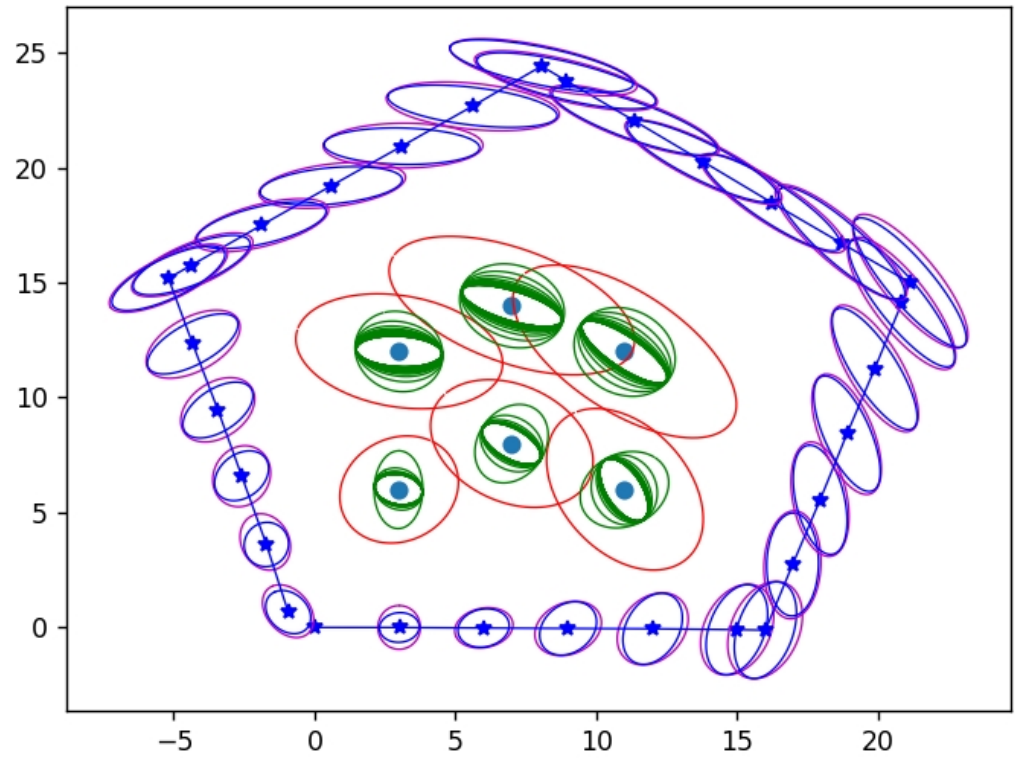
The ellipses don't become more "spherical" because we linearize the update function to compute the covariance.

$0.1 \times \sigma_\beta$:



This has the opposite effect.

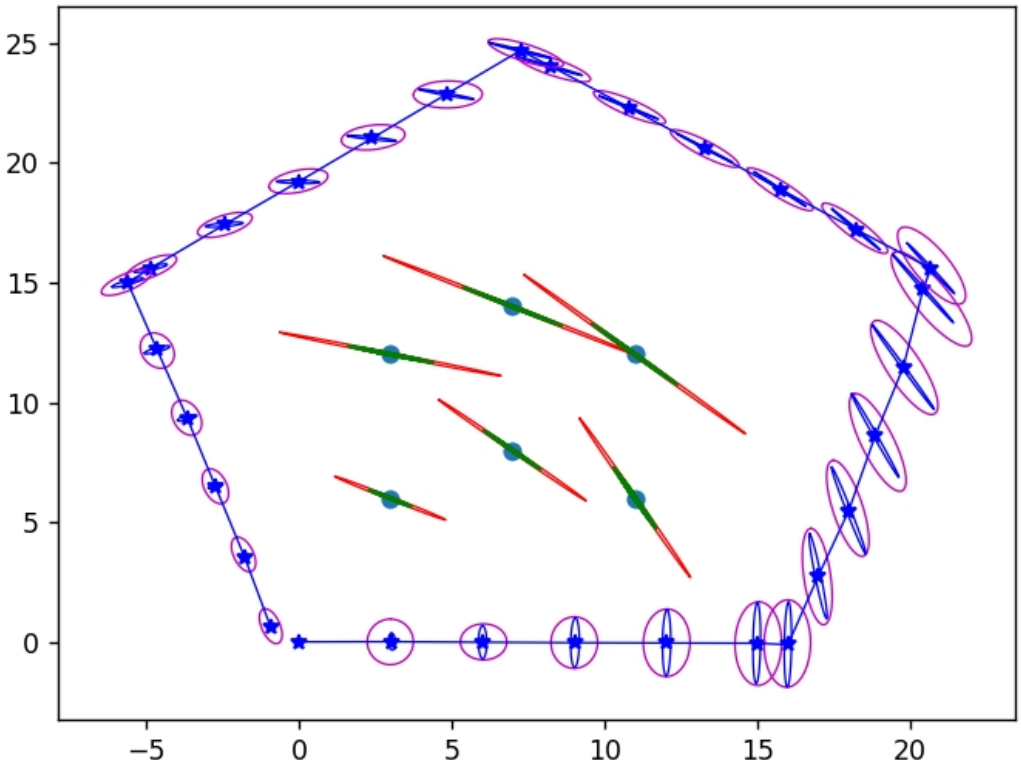
$10 \times \sigma_r$:



σ_r controls the variance of the distances from the robot to the landmarks. When it increases, the ellipses around the landmarks become larger and more "spherical". This is because the system is more uncertain about the landmarks'

distances to the robot. And in turn, the ellipses around the robot also become larger.

$0.1 \times \sigma_r$:



This has the opposite effect.

3. 1. When the robot does not receive valid measurement of a landmark, ignore that landmark and does not update its position and covariance. This is very useful when the robot is scanning some large scenes when the robot is unable to see all the landmarks. It is meaningless to update those distant landmarks or use them to update the robot's location.

   2. Group landmarks into blocks according to their positions. Perform block updating instead of updating each landmark individually.

   3. Use key frames. Only update everything during key frames. During non key frames, do a subset of the SLAM tasks. E.g. during key frames both update the map and localize the robot, and during non key frames only localize the robot.

4 Code Submission

Upload your `ekf_slam.py`. Please do not upload the `data/` folder or any other data or image.